GIT Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework #06 Report

Muhammed Sefa Cahyir
1801042686

1. SYSTEM REQUIREMENTS

No need expensive system for this project. If don’t use with big datas.

It is enough to have a Java and a machine to run JVM

The minimum system requirements for Java Virtual Machine are as follows:

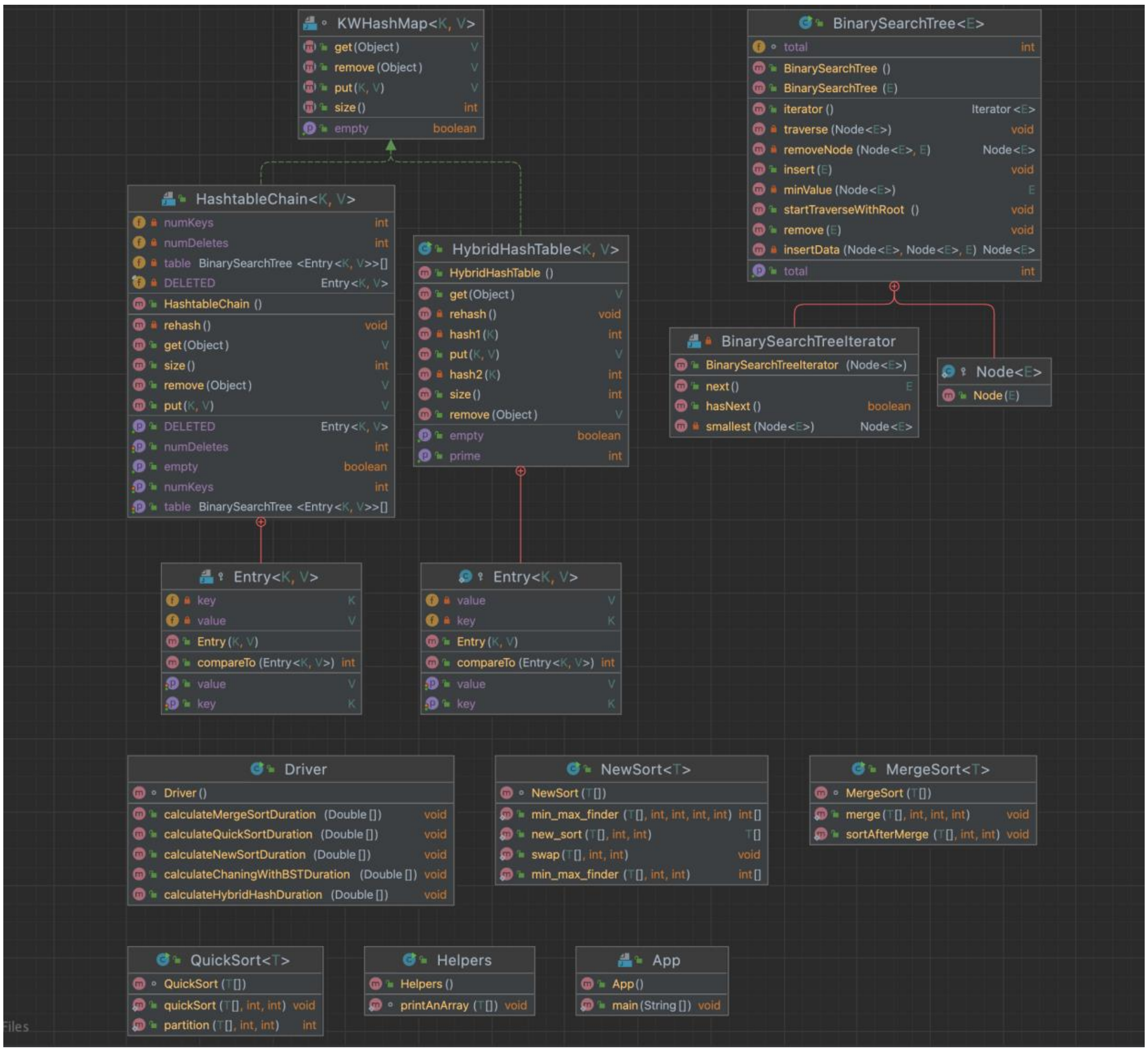
Windows 8/7/Vista/XP/2000.

Windows Server 2008/2003.

Intel and 100% compatible processors are supported.

Pentium 166 MHz or faster processor with at least 64 MB of physical RAM. 98 MB of free disk space.

2. USE CASE AND CLASS DIAGRAMS



3. OTHER DIAGRAMS

No other diagram

4. PROBLEM SOLUTION APPROACH

In this homework Q1 part1 problem is making HashChain table with BST, we already make OpenHashTable with array and HashChain table with linked list. But in first time we are used HashChain table with BST.

For using this we need to extend Iterable and Comperable. We had add Iterable for searching a key in BST with iterate it and later we used Comparable for compare this and try to find right solution.

Q1 part2 problem is was combinate 2 hash technique and use these together, in this we learned how is working double hashing and coalesced hashing, double hashing is helping us when there is a collision or without some case we are hashing key 2 times with a formula, it make it more safe for password or key hiding and we don’t need to think about collision again.
And other hashing method is coalesced hashing, when there is a collision store somewhere else and store the where is stored the next value, when making this we can find the next value easily.

In Q2 for this homework, our problem is use different sorting algorithms and create a new sorting algorithms and compare this with different size.

First sorting algorithm is quick sort, for sorting data with find middle data and sorting for small and bigger than this data. It is like divide and conquer algorithm.

Second sorting algorithm is merge sort, for sorting data with divide until 2 item in every part and sorting this and later merge these data’s. It is another a divide and conquer algorithm but these is useful for trees.

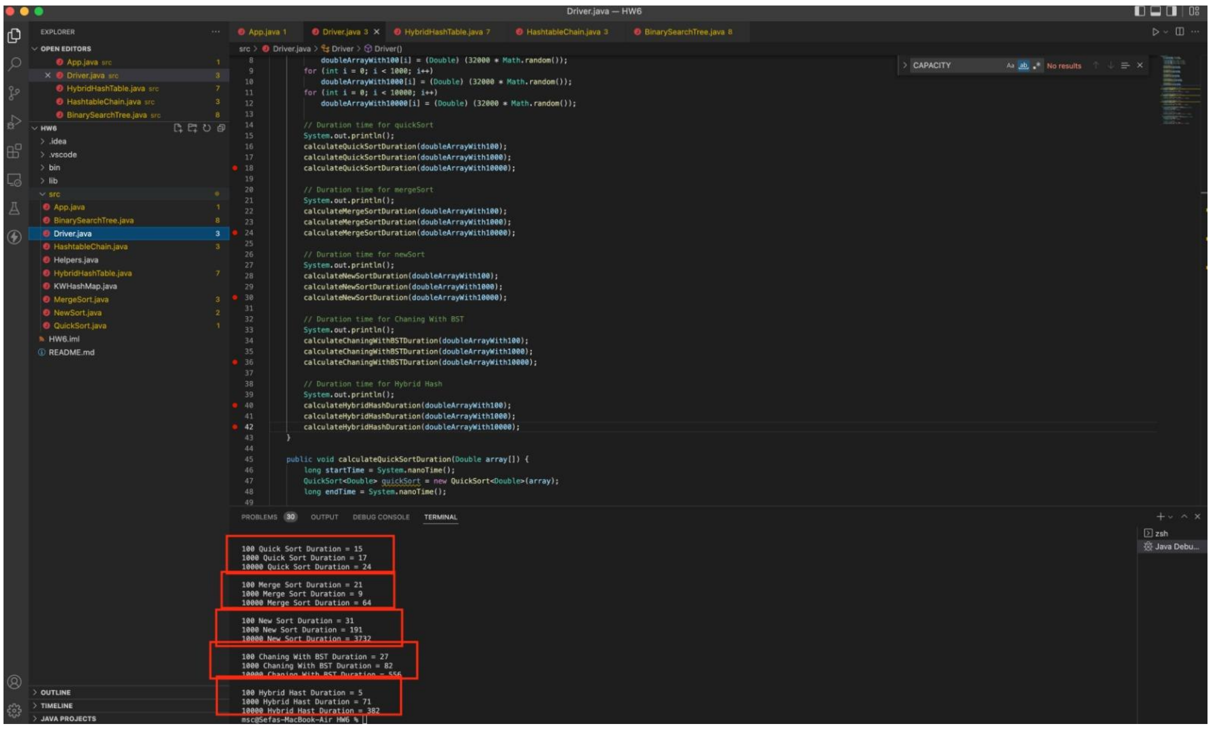
Third sorting algorithm is new sort, for sorting data with compare first and last data one by one, and it is useful for arrays. Cheking every small and bigger value in list or array and exchange it.

5. TEST CASES

Q1 is tested with diffirent variations like add, remove, get and add with become collision. And it is tested with 100,1000,10000 size double arrays and showed the running time.

Q2 is tested with diffirent data type like String, Integer, Double and Double conclusion is showed.

6. RUNNING AND RESULTS



```java
            doubleArrayWith100[i] = (Double) (32000 * Math.random());
        for (int i = 0; i < 1000; i++)
            doubleArrayWith1000[i] = (Double) (32000 * Math.random());
        for (int i = 0; i < 10000; i++)
            doubleArrayWith10000[i] = (Double) (32000 * Math.random());

        // Duration time for quickSort
        System.out.println();
        calculateQuickSortDuration(doubleArrayWith100);
        calculateQuickSortDuration(doubleArrayWith1000);
        calculateQuickSortDuration(doubleArrayWith10000);

        // Duration time for mergeSort
        System.out.println();
        calculateMergeSortDuration(doubleArrayWith100);
        calculateMergeSortDuration(doubleArrayWith1000);
        calculateMergeSortDuration(doubleArrayWith10000);

        // Duration time for newSort
        System.out.println();
        calculateNewSortDuration(doubleArrayWith100);
        calculateNewSortDuration(doubleArrayWith1000);
        calculateNewSortDuration(doubleArrayWith10000);

        // Duration time for Chaning With BST
        System.out.println();
        calculateChaningWithBSTDuration(doubleArrayWith100);
        calculateChaningWithBSTDuration(doubleArrayWith1000);
        calculateChaningWithBSTDuration(doubleArrayWith10000);

        // Duration time for Hybrid Hash
        System.out.println();
        calculateHybridHashDuration(doubleArrayWith100);
        calculateHybridHashDuration(doubleArrayWith1000);
        calculateHybridHashDuration(doubleArrayWith10000);
    }

    public void calculateQuickSortDuration(Double array[]) {
        long startTime = System.nanoTime();
        QuickSort<Double> quickSort = new QuickSort<Double>(array);
        long endTime = System.nanoTime();
```

```
100 Quick Sort Duration = 15
1000 Quick Sort Duration = 17
10000 Quick Sort Duration = 24

100 Merge Sort Duration = 21
1000 Merge Sort Duration = 9
10000 Merge Sort Duration = 64

100 New Sort Duration = 31
1000 New Sort Duration = 191
10000 New Sort Duration = 3732

100 Chaning With BST Duration = 27
1000 Chaning With BST Duration = 82
10000 Chaning With BST Duration = 556

100 Hybrid Hast Duration = 5
1000 Hybrid Hast Duration = 71
10000 Hybrid Hast Duration = 382
msc@Sefas-MacBook-Air HW6 % 
```